IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR PATENT

DYNAMIC DATABASE MANAGEMENT SYSTEM AND METHOD

Inventor: Hovhannes Ghukasyan,
Souren L. Chilingarian, and
Yervant D. Lepejian

FIELD OF THE INVENTION

The present invention generally relates to techniques for managing modifications to databases and in particular, to a dynamic database management system and method.

BACKGROUND OF THE INVENTION

One of the most important features of a database is that it provides a certain degree of isolation between its data structures and software that uses them. Unfortunately, this degree of isolation is often low in the real world, especially in cases where the system architecture is not specifically designed to support future modifications to the database schema. In such cases, expensive software modifications that effectively drop the system's usability to zero may result.

The most common and simplest example of a database modification is adding an attribute to a table. Even this simple example, however, normally requires: altering the table design to add the attribute, and modifying the front-end software so that it gives users access to the new attribute (unless the attribute names are retrieved from the database and they are self-explanatory). In a more complex example, the new attribute may belong to a relationship that does not exist in the database. In this case, the modifications will involve creation of a new table and significant changes to the software to make use of the newly created table.

Although certain conventional approaches generally work fine for adding or deleting attributes of objects or existing relationships without modifying the software that uses the database, they generally do not support transparent adding of a new attribute if the table representing the relationship does not already exist in the database.

OBJECTS AND SUMMARY OF THE INVENTION

Accordingly, an object of the present invention is to provide a dynamic database management system and method.

Another object of the present invention is to provide a dynamic database management system and method that automatically imports data associated with new attributes into a database.

Another object of the present invention is to provide a dynamic database management system and method that automatically imports data associated with new attributes into a database, and makes them immediately available to end-users or computer programs using the database.

Still another object of the present invention is to provide a dynamic database management system and method that automatically imports data associated with new attributes into a database by creating new tables as necessary, and makes the imported data immediately available to end-users or computer programs using the database.

These and additional objects are accomplished by the various aspects of the present invention, wherein briefly stated, one aspect is a dynamic database management system comprising: a data dictionary including identifications of related groups of tables in a database, information of tables in the related groups, and identifications of parameters of the related groups; and a data importer receiving an input including data to be imported into the database, an indication of one of the related groups that is associated with the data, and indications of parameters associated with the data, wherein the data importer appends one or more portions of the data associated with existing parameters to corresponding one or more existing tables associated with the existing parameters and having tables of the indicated one of the related groups as references, appends data associated with new parameters to a new table created for the new parameters, and updates the data dictionary to include the identifications and information of the new table and new parameters.

In another aspect, a method for managing a dynamic database comprises: receiving an input including data to be imported into a database, an indication of a related group of tables that is associated with the data, and indications of parameters associated with the data; forming a set of existing parameters and a set of new parameters from the parameters associated with the data, based upon parameter information stored in a data dictionary for the related group of tables; appending one or more portions of the data associated with the set of existing parameters to corresponding one or more existing tables in the database and having tables of the related group as references; importing a remaining portion of the data associated with the set of new parameters to a new table created for the new parameters; and updating information in the data dictionary to include identifications and information of the new table and the new parameters.

Additional objects, features and advantages of the various aspects of the invention will become apparent from the following description of its preferred embodiments, which description should be taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates, as an example, a block diagram of a dynamic database management system, utilizing aspects of the present invention.

FIG. 2 illustrates, as an example, a structure diagram for a data dictionary, utilizing aspects of the present invention.

FIG. 3 illustrates, as an example, a structure diagram for an input file, utilizing aspects of the present invention.

FIG. 4 illustrates, as an example, a flow diagram of a method for dynamic database management, utilizing aspects of the present invention.

FIG. 5 illustrates, as an example, a dependency graph depicting relationships between tables for a database schema.

FIG. 6 illustrates, as an example, a reference group table in a data dictionary, utilizing aspects of the present invention.

FIG. 7 illustrates, as an example, a references table in a data dictionary, utilizing aspects of the present invention.

FIG. 8 illustrates, as an example, a parameters table in a data dictionary, utilizing aspects of the present invention.

FIG. 9 illustrates, as an example, a folders table in a data dictionary, utilizing aspects of the present invention.

FIG. 10 illustrates, as an example, a parameter to folder mapping table in a data dictionary, utilizing aspects of the present invention.

FIG. 11 illustrates, as an example, an input file to the database management system, utilizing aspects of the present invention.

FIG. 12 illustrates, as an example, an update record to the parameters table, utilizing aspects of the present invention.

FIG. 13 illustrates, as an example, update records to the folders table, utilizing aspects of the present invention.

FIG. 14 illustrates, as an example, an update record to the parameter to folder mapping table, utilizing aspects of the present invention.

FIG. 15 illustrates, as an example, the parameter tree resulting from the database management system, utilizing aspects of the present invention, processing the input file of **FIG. 11**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 illustrates, as an example, a block diagram of a dynamic database management system designed to support databases subject to frequent schema modifications. Included in the system are a data dictionary **101**, a data importer **102** and a query front-end **103**. Together, these modules provide the ability to automatically import data included in an input file and associated with new attributes not previously included in the data dictionary **101** to a database **104**, and make the data available to end-users for database queries through the query front-end **103**. The modules may be implemented in any one of several well-known ways, including software, firmware, hard-wired logic, or combinations thereof, in conjunction with a processor as appropriate.

The data dictionary **101** stores metadata including table relationship information and attribute locations. The data importer **102** writes data into the database **104** based on the information stored in the data dictionary **101**, and updates the data dictionary as necessary after such writing. The query front-end **103** presents the database **104** to end-users in the form of a parameter tree with user-defined folders and items that correspond to the table attributes. Usage of the parameter tree as a user front-end addresses two issues. First, the end-user works with the names that are familiar to him or her, organized into a custom folder structure adopted to the domain and/or usage of the data instead of dealing with the database directly, where the naming and organization of the data can be quite cryptic to the ordinary end-user. Second, the dynamically added parameters are immediately made available to the end-user and can be then moved to the appropriate folders by the database administrator or other authorized user.

FIG. 2 illustrates, as an example, a diagram detailing the structure of the data dictionary **101**. A RefGroup table **201** contains a record per object or table relationship, and has columns RefGroupID **206** and RefGroupName **207**. Objects or table relationships in this system are allowed to have detached attributes. A

References table **202** contains names of the reference tables and fields that should be used to link to the detached attributes of a given object or table relationship. The References table **202** has columns RefName **208**, RefGroupID **209**, RefTable **210**, IDField **211** and NameField **212**. A Parameters table **203** contains names of data items that are used in the Query Front-End **103**, and their types and definitions. Definitions are stored in the form *TableName.FieldName*. This table has columns ParamID **213**, ParamName **214**, ParamType **215**, ParamDefinition **216**, and RefGrouptID **217**. Records in the RefGroup table **201**, References table **202** and Parameters table **203** are linked through their RefGroupID fields, **206**, **209** and **217**, respectively. A Folders table **204** contains information of user-defined folders used in the Query Front-End **103**. This table has columns FolderID **218**, FolderName **219** and Parent **220**. Folders are displayed to the end-user in a parameter tree fashion. A ParamFolders table **205** provides the necessary information for parameter-to-folder mapping. This table contains columns FolderID **221** and ParamID **222**.

FIG. 3 illustrates, as an example, a diagram detailing the structure of an input file **300**. The input file **300** contains one or more sections such as sections **301**, **302** and **303**. Each section, such as exemplified by section **301**, contains a RefGroupName field **304** for providing the name of the reference group to which the enclosed data belongs, and a data table **305**. The name provided in the RefGroupName field **304** should match one of the reference group names provided in the RefGroupName column **207** of the RefGroup table **201** in the Data Dictionary **101**. The data table **305** contains a header portion **306** and a body portion **307**. The header portion **306** includes the names of references, such as RefName(1) **309**, RefName(2) **310** and RefName(3) **311**, from the References table **202** linked through the same RefGroupID to the reference group indicated in the RefGroupName field **304**. The header portion **306** also contains at least one parameter name associated with the reference group accompanied by its data type, such as ParamName(1) **312** and ParamType(1) **313**. The body portion **307** contains the data or values to be imported.

FIG. 4 illustrates, as an example, a flow diagram of a method for dynamic database management performed by the data importer **102**. In **401**, an input file is received. The input file is then checked to make sure that its structure conforms to that described in reference to **FIG. 3**. Assuming it is valid, then in **402**, the reference group name associated with the data to be imported is read from the RefGroupName field **304** of a section of the input file **300** being processed. In **403**, references corresponding to the reference group name are then retrieved from the References table **202** of the Data Dictionary **101**.

In **404**, a parameters set is created from the retrieved references and column information in the data table **305** of the input file **300**, and split into existing and new parameters sets based on the existence of parameters in the Parameters table **203** of the Data Dictionary **101**. In **405**, one or more existing tables for the existing parameters are then identified from information stored in the Parameters table **203** of the Data Dictionary **101**. For example, if the existing parameter set is {x, y, z} and the Parameters table **203** discloses a table A having the indicated reference group as references and parameters x and y as attributes and another table B also having the indicated reference group as references and parameter z as an attribute, then tables A and B will be identified in this case as the one or more existing tables referred to above. In **406**, data associated with the existing parameters are then imported into one or more temporary tables or files created to correspond to the one or more existing tables according to common existing parameters, and then the one or more temporary tables or fields are appended to their corresponding one or more existing tables. In **407**, a new table name is created for the data associated with the new parameters. In **408**, the Parameters table **203** of the Data Dictionary **101** is updated to include the new parameters and new table. In **409**, the new table is created, and in **410**, the data associated with the new parameters are imported into the new table. Finally, in **411**, the Folders table **204** and ParamFolders table **205** are updated to respectively add a new folder for the new table and mapping information from the new table to the new parameters.

A simple example is now described to further illustrate the dynamic database management system and method embodying the present invention. In **FIG. 5**, a dependency graph **500** is illustrated as depicting relationships between tables for a database schema, and tables in the data dictionary corresponding to the database schema are illustrated in **FIGS. 6~10**. In **FIG. 11**, an example of an input file to the database management system is illustrated.

According to the method described in reference to **FIG. 4**, after receiving the input file depicted in **FIG. 11**, the data importer **102** reads the reference group name from the RefGroupName field of the input file. In this example, the reference group is named DefectClassData. According to record **608** of the RefGroup table depicted in **FIG. 6**, the reference group named DefectClassData has a RefGroupID of

"7". Based upon this value for RefGroupID, the data importer **102** then retrieves the reference names for the reference group from the References table, which according to records **712** and **713** of the References table depicted in **Fig. 7**, are "Wafer" and "DefectClass". This can also be readily seen from the dependency graph **500** of **FIG. 5**, wherein the directions of the arrows indicate that the table DEFECTCLASSDATA **513** is dependent upon reference tables WAFER **503** and DEFECTCLASS **508**.

Defining R as the set of reference names for the references, then R = {"Wafer", "DefectClass"}. Likewise, defining H as the set of columns from the input file, then H = {"Wafer", "DefectClass", Dcd1, Dcd2, Dcd3} according to the input file depicted in **FIG. 11**.

A parameter set P may then be determined as P = H - R, which in this case is equal to {Dcd1, Dcd2, Dcd3}. This set is then split based on the Parameters table depicted in **FIG. 8**, into an existing parameter set, PE = {Dcd1, Dcd2}, and a new parameter set PN = {Dcd3}. As is evident from the ParamDefinition fields in records **805** and **806** in the Parameters table depicted in **FIG. 8**, both existing parameters, Dcd1 and Dcd2, are associated with the same table, DefectClassData. Accordingly, the data importer **102** creates a temporary table or file having columns R + PE ("Wafer", "DefectClass", Dcd1, Dcd2), stores data corresponding to the existing parameters from the input file to the temporary table, and appends the temporary table to the existing DefectClassData table having the same columns.

For the new parameter, Dcd3, the data importer **102** first generates a new table name, such as DCData1, and then creates a new record, such as record **1201** depicted in **FIG. 12** for the new parameter, for inclusion in the Parameters table. The data importer **102** then creates the new table named DCData1 and having columns R + NE ("Wafer", "DefectClass", Dcd3), and imports data corresponding to the new parameters from the input file into the newly created table. The new table is then added to the dependency graph **500** by connecting an arrow pointing from the table WAFER **503** to the new table DCData1 and another arrow pointing from the table DEFECTCLASS **508** to the new table DCData1.

Finally, to make the data available to end-users, the data importer **102** creates new folders named "New Parameters" and "DefectClassData" as depicted by records **1301** and **1302** in an update to the Folders table as depicted in **FIG. 13**, and associates the new parameter to the new folder "DefectClassData" as depicted by record **1401** in an update to the ParamFolders table as depicted in **FIG. 14**. The resulting parameter tree would then appear as depicted in **FIG. 15**, as presented to the end-user by the Query Front-End **103**.

Next time the data importer **102** gets a similar input file, it will find from the Parameters table **203** that all three parameters, Dcd1, Dcd2 and Dcd3, exist in the database, but the first two, Dcd1 and Dcd2, are in the DefectClassData table and the last one, Dcd3, is in the DCData1 table. The data importer **102** will then split data from the input table into two sub-tables based on that split of the existing parameters, and append the first one to the DefectClassData table and the second one to the DCData1 table.

Although the various aspects of the present invention have been described with respect to a preferred embodiment, it will be understood that the invention is entitled to full protection within the full scope of the appended claims.